Introduction to Model-Checking
Part 4— Still Modeling

Exercise 1. Token ring

The goal is to model a token-ring network with K sites, where K is a parameter of the problem. A token ring is a specific networking configuration in which you have K processors, or *site*, arranged in a cycle (the topology is a ring; each processor is only linked to its unique successor and predecessor). We use a "token" to solve a mutual exclusion problem. Only the processor with the token can send a message in the network. The other processor can still compute but cannot communicate.

Each site is composed of a station and a client. The station is in charge of circulating the token.

The client number i has three possible states

- **idle** client i does nothing
- **wait** client wants to do something and is waiting for the token
- **cs** client is in the critical section

The token has 2.K possible states, that is k instances of the following states with $i \in 1..K$.

- **token.i** token is available at site i
- **after.i** token is in transit between site i and i+1

The following net gives a description of one of the site and a part of the ring states.

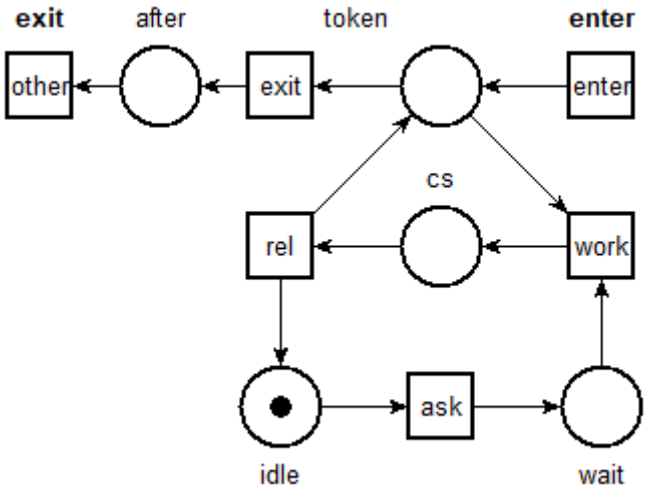

Figure 1. net for a single site (file mono.ndr)

Question 1. Edit a copy of this net and take care of using the right labels. Then build a network with four sites by composing four copies of this net using the tpn format. Be careful that you need to rename the labels enter and exit so that you obtain the right topology; the exit transition of site 2

(other.2) should be the same than the enter transition of site 3 (enter.3). See the manual page for tina for a description of the tpn format (http://projects.laas.fr/tina/manuals/formats.html).

For example, you could use a tpn script of the form:

```
# we load a copy of mono.ndr and put a token in the start place
load mono.ndr
pl token(1)
ren after1/exit
# we push new copies of mono on "the stack"
# and rename enter/exit to glue them correctly
load mono.ndr
ren after1/enter after2/exit
# you can continue
…
ren after3/enter enter/exit
sync 4
```

You can draw the resulting net with neato and check that everything is all right.

Question 2. Simulate the resulting net with the stepper and find possible problems in the models. You should find that this solution does not prevent starvation: a client may be willing to work (in place wait) but never enter critical section.

Question 3. Could you identify two main reasons that may lead to starvation? Find a scenario for each of these two cases.

Question 4. Propose a solution to correct this problem using inhibitor arcs and/or priorities if necessary. You can do it directly on the ndr file you have produced (keep a backup) and you only need to do it on a single site.

Question5. We give a possible solution in Figure 2 (we just show it for site 1). Could you explain what are the meaning of transitions t3 and t4; likewise for places nbMax and frozen.

Could you find an equivalent solution without inhibitor arcs?

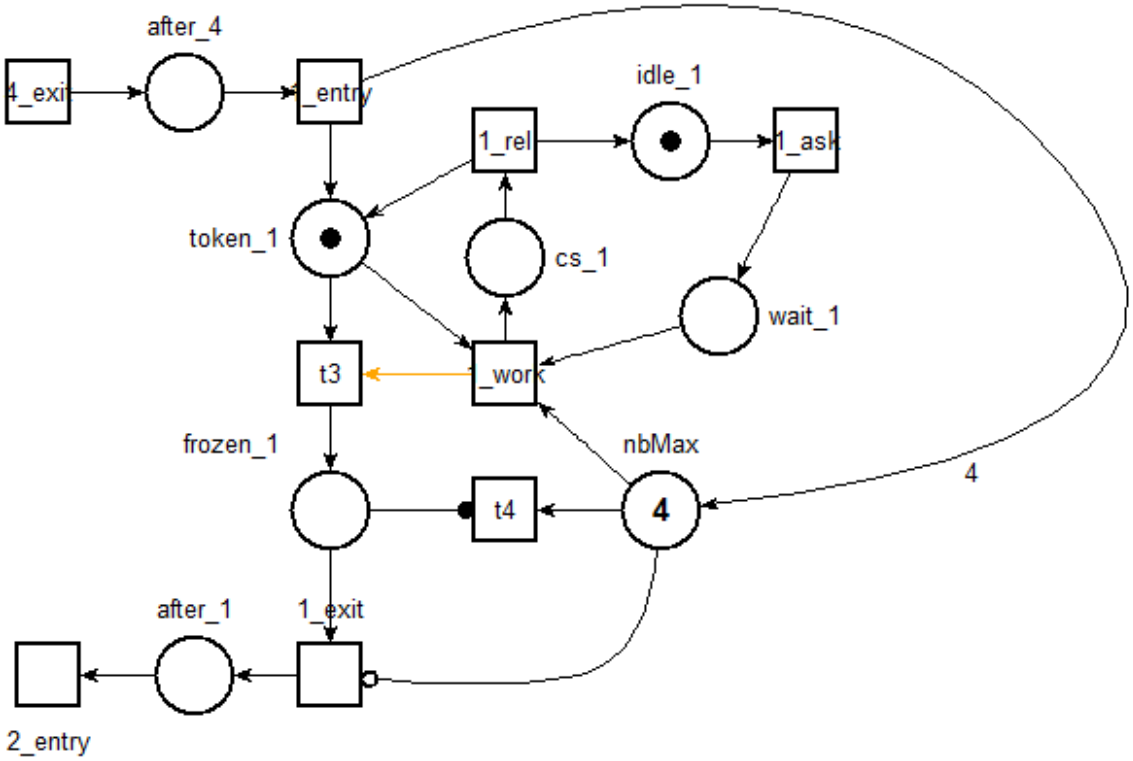

Figure 2 a proposal for correcting our model, exercise 1 question 5

Exercise 2. Dining philosophers

N philosophers are seated around a table in front of a plate of spaghettis. They can be in three possible states (thinking, waiting or eating). Each philosopher needs two forks to eat, one in each hand, but there is only one fork for each plate. So they need to take the fork to their left and the one to their right if they want to eat. These actions cannot be done at the same time, but the order is not fixed. This mean that two philosophers seated next to each other should never be able to eat at the same time.

Take inspiration from the tpn script used in the previous exercise to model a system with N=4 philosophers. We will make different versions of increasing complexity.

Question 1. We consider a simplified version of the problem in which all the free forks can be found at the center of the tables. So you have a place, with N tokens initially, that models the number of free forks. The forks still need to be taken sequentially.

Question 2. Try to model the exact system, with exactly one fork between two adjacent philosophers.